Факультет компьютерных систем и сетей

Кафедра информатики

Дисциплина: Объектно-ориентированное программирование (ООП)

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту на тему

«Онлайн библиотека»

Выполнил: студент гр.753501 Снегур А.Н.

Руководитель: ассистент кафедры информатики Рогов М.Г.

**Содержание**

[Введение 3](#_Toc415311770)

[1. Постановка задачи 4](#_Toc415311771)

[1.1. Описание предметной области 4](#_Toc415311772)

[1.2. Задачи проектирования 5](#_Toc415311773)

[1.3. Требования к программному обеспечению 6](#_Toc415311774)

[2. Проектирование задачи 7](#_Toc415311775)

[2.1. Проектирование архитектуры приложения 7](#_Toc415311776)

[2.2. Используемые технологии 8](#_Toc415311777)

[2.3. Среды реализации задачи 8](#_Toc415311778)

[2.3.1. JetBrains PyCharm 8](#_Toc415311779)

[2.3.2. Python 9](#_Toc415311780)

[2.3.3. MongoDB Server 10](#_Toc415311781)

[3. Программная реализация 11](#_Toc415311786)

[3.1. Физическая структура приложения 11](#_Toc415311787)

[3.2. Архитектура приложения 12](#_Toc415311787)

[4. Описание применения 18](#_Toc415311788)

[4.1. Main 18](#_Toc415311787)

[4.2. Catalog 19](#_Toc415311787)

[4.3. Account/Registration 20](#_Toc415311787)

[4.4. Info 21](#_Toc415311787)

[4.5. Transfer/Statistics 21](#_Toc415311787)

[Заключение 22](#_Toc415311795)

[Список использованных источников 23](#_Toc415311796)

[Приложения 24](#_Toc415311797)

# Введение

Электронная библиотека — упорядоченная коллекция разнородных электронных документов (в частности книг). Может быть [веб-сайтом](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D0%B0%D0%B9%D1%82), где постепенно накапливаются различные [тексты](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D0%BA%D1%81%D1%82) (чаще литературные, но также научные и любые другие, вплоть до компьютерных программ) и медиафайлы, каждый из которых самодостаточен и в любой момент может быть востребован читателем. Электронные библиотеки могут быть универсальными, стремящимися к наиболее широкому выбору материала, и более специализированными, как [Фундаментальная электронная библиотека](https://ru.wikipedia.org/wiki/%D0%A4%D1%83%D0%BD%D0%B4%D0%B0%D0%BC%D0%B5%D0%BD%D1%82%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D1%8D%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D0%B1%D0%B8%D0%B1%D0%BB%D0%B8%D0%BE%D1%82%D0%B5%D0%BA%D0%B0) или проект [Сетевая Словесность](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%82%D0%B5%D0%B2%D0%B0%D1%8F_%D0%A1%D0%BB%D0%BE%D0%B2%D0%B5%D1%81%D0%BD%D0%BE%D1%81%D1%82%D1%8C), нацеленный на собирание авторов и типов текста, наиболее ярко заявляющих о себе именно в [Интернете](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D0%BD%D0%B5%D1%82). Возникло большое число избыточной информации, в которой иногда трудно сориентироваться и выбрать нужные сведения.

Конечно, должно быть программное обеспечение, работающее с данным видом предметной области, которое, в свою очередь, состоит из множества модулей, и основным из них является, конечно же, база данных имеющихся книг.

Так как в наше время все получают информацию в основном из своих гаджетов, никто просто так не пойдет в библиотеку, чтобы взять интересующие его книги: проще найти их в интернете. Но поиск в интернете не всегда бывает безопасным, и именно поэтому я решил написать специальное приложение, помогающее пользователю безопасно находить и приобретать интересующую его литературу.

# Постановка задачи

## Описание предметной области

Библиотека  — [учреждение](https://ru.wikipedia.org/wiki/%D0%A3%D1%87%D1%80%D0%B5%D0%B6%D0%B4%D0%B5%D0%BD%D0%B8%D0%B5), собирающее и хранящее произведения печати и [письменности](https://ru.wikipedia.org/wiki/%D0%9F%D0%B8%D1%81%D1%8C%D0%BC%D0%B5%D0%BD%D0%BD%D0%BE%D1%81%D1%82%D1%8C) для общественного пользования, а также осуществляющее справочно-библиографическую работу. В настоящее время всё более распространяются и входят в [фонд](https://ru.wikipedia.org/wiki/%D0%A4%D0%BE%D0%BD%D0%B4)библиотеки [микрофиши](https://ru.wikipedia.org/wiki/%D0%9C%D0%B8%D0%BA%D1%80%D0%BE%D1%84%D0%B8%D1%88%D0%B0), [микрофильмы](https://ru.wikipedia.org/wiki/%D0%9C%D0%B8%D0%BA%D1%80%D0%BE%D1%84%D0%B8%D0%BB%D1%8C%D0%BC), [диапозитивы](https://ru.wikipedia.org/wiki/%D0%94%D0%B8%D0%B0%D0%BF%D0%BE%D0%B7%D0%B8%D1%82%D0%B8%D0%B2), [аудио](https://ru.wikipedia.org/wiki/%D0%90%D1%83%D0%B4%D0%B8%D0%BE%D0%BA%D0%B0%D1%81%D1%81%D0%B5%D1%82%D0%B0) и [видеокассеты](https://ru.wikipedia.org/wiki/%D0%92%D0%B8%D0%B4%D0%B5%D0%BE%D0%BA%D0%B0%D1%81%D1%81%D0%B5%D1%82%D0%B0), также всё более широкое распространение получают [электронные носители](https://ru.wikipedia.org/wiki/%D0%9D%D0%BE%D1%81%D0%B8%D1%82%D0%B5%D0%BB%D1%8C_%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%86%D0%B8%D0%B8#_%D0%9A%D0%BB%D0%B0%D1%81%D1%81%D0%B8%D1%84%D0%B8%D0%BA%D0%B0%D1%86%D0%B8%D1%8F_%D0%BD%D0%BE%D1%81%D0%B8%D1%82%D0%B5%D0%BB%D0%B5%D0%B9) (CD-ROM, DVD-ROM).

Деятельность библиотек по обслуживанию читателей осуществляется в двух основных формах. Библиотечный [абонемент](https://ru.wikipedia.org/wiki/%D0%90%D0%B1%D0%BE%D0%BD%D0%B5%D0%BC%D0%B5%D0%BD%D1%82) предоставляет читателю право получить издание из библиотеки в своё полное распоряжение на определённый срок. В другом случае читатель имеет возможность знакомиться с книгой только в помещении библиотеки (как правило, в специально отведённом читальном зале).

При регистрации в библиотеке пользователь должен указать свои данные для дальнейшего сотрудничества с ней. В некоторых случаях это имя, фамилия, место жительства, а в некоторых - электронная почта, личный пароль и имя пользователя.

Библиотека создается с целью обогащения человека знаниями, и она является бесплатным источником их получения. Однако некоторые читатели могут иметь ряд привилегий за счет каких-либо определенных действий.

Стоит так же отметить, что в библиотеке всегда ведется статистика различных данных, а именно:

* Общее число зарегистрированных пользователей;
* Названия самых популярных на текущий момент времени книг или авторов;
* Прочее (общее количество выданных книг, число задолженностей, выдача книг в какой-то конкретный день и т.д.);

## Задачи проектирования

Чтобы приступить к непосредственному проектированию библиотеки, необходимо выделить минимальный набор основных сущностей и указать типы связи между ними.

В рассматриваемой предметной области можно выделить следующие основные сущности:

Пользователь – содержит информацию о нике, личному кошельку, списку приобретенных книг, персональному изображению пользователя.

Администратор – привилегированный пользователь. Имеет доступ к статистике и переводу коинов.

Книга – содержит информацию о названии, авторе, цене книги.

Статистика – содержит информацию об общем количестве приобретенных книг, заработанных денег и общему числу пользователей.

Каталог – содержит информацию о книгах и заказах на них.

Заказ – информация о заказе книги.

Перевод – перевод коинов пользователям.

Типы связей между сущностями.

1:1:

Пользователь – Пользователь (детали).

1:n:

Администратор – Статистика

Администратор – Перевод

Пользователь – Каталог

Каталог – Книга

Пользователь – Книга

Пользователь – Заказ

Пользователь – Аккаунт

m:n:

Администратор – Каталог

Администратор – Аккаунт

## Требования к разрабатываемому программному обеспечению

Готовый программный продукт должен удовлетворять следующим требованиям:

1. Иметь простой и понятный интерфейс;
2. Доступ к базе данных через приложение должен осуществляться посредством технологии Mongo Database (NoSQL Database);
3. Использовать систему ролей для пользователей:
   1. Администратор (имеет доступ ко всем модулям приложения и неограниченное количество коинов);
   2. Пользователь (рядовой читатель со стандартным количеством коинов);
4. Позволять легко создавать/изменять данные о читателях;
5. Предоставлять актуальную информацию о купленных книгах, общему числу пользователей и общему количеству заработанных коинов;
6. Позволять отслеживать информацию об изменениях информации у каждого пользователя;
7. Не должно иметь большой отклик при загрузке больших данных;
8. Не должно иметь багов при покупке и отслеживании статистики;
9. Сопровождение звуковыми сигналами при входе и выходе для подтверждения успеха/провала операции;
10. Широкая вариация страничек пользователей для создания своей собственной уникальной странички;

Таким образом, задача данного курсового проекта сводится к разработке приложения для обеспечения работы онлайн библиотеки. Готовое приложение должно иметь понятный интерфейс и быть удобным в использовании.

# Проектирование задачи

## Проектирование архитектуры приложения

Для разработки проекта потребовалось составить схему на языке UML.

На рис.1 представлена схема архитектуры основных взаимосвязей приложения. На ней можно видеть сущности и типы связей.

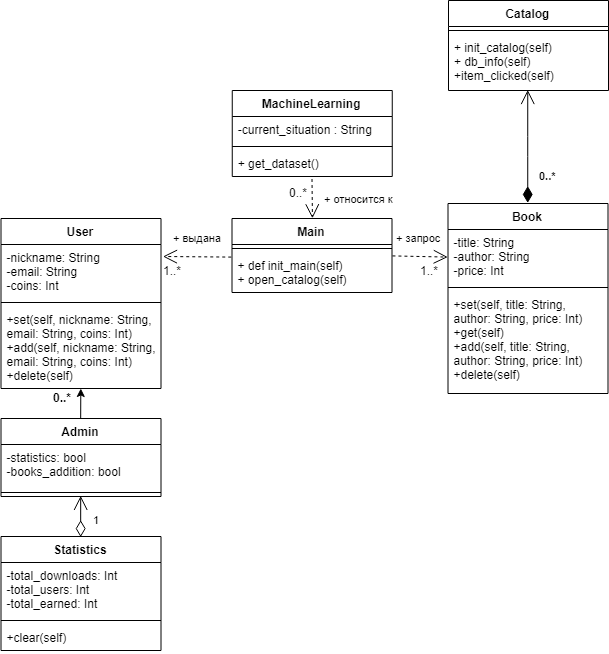


Рис. 1. Схема архитектуры основных взаимосвязей приложения

# Используемые технологии

Приложение реализовано на языке Python с использованием технологий Python Tkinter. В качестве среды разработки был выбран JetBrains PyCharm. Для создания и работы с базой данных была использована Mongo DataBase. Все запросы написаны на NoSQL.

## Среды реализации задачи

### JetBrains PyCharm

PyCharm — [интегрированная среда разработки](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D0%B3%D1%80%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D1%80%D0%B5%D0%B4%D0%B0_%D1%80%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B8) для языка программирования [Python](https://ru.wikipedia.org/wiki/Python" \o "Python). Предоставляет средства для анализа кода, графический отладчик, инструмент для запуска юнит-тестов и поддерживает веб-разработку на [Django](https://ru.wikipedia.org/wiki/Django" \o "Django). PyCharm разработана компанией [JetBrains](https://ru.wikipedia.org/wiki/JetBrains" \o "JetBrains) на основе [IntelliJ IDEA](https://ru.wikipedia.org/wiki/IntelliJ_IDEA" \o "IntelliJ IDEA).

PyCharm работает под операционными системами [Windows](https://ru.wikipedia.org/wiki/Windows" \o "Windows), [Mac OS X](https://ru.wikipedia.org/wiki/Mac_OS_X" \o "Mac OS X) и [Linux](https://ru.wikipedia.org/wiki/Linux" \o "Linux).

PyCharm был выпущен на рынок интегрированных сред разработки для Python для создания конкуренции с [PyDev](https://ru.wikipedia.org/w/index.php?title=PyDev&action=edit&redlink=1" \o "PyDev (страница отсутствует)) и более распространённой среды разработки [Komodo IDE](https://ru.wikipedia.org/wiki/ActiveState_Komodo" \o "ActiveState Komodo). Бета-версия была выпущена в [июле](https://ru.wikipedia.org/wiki/%D0%98%D1%8E%D0%BB%D1%8C) [2010 года](https://ru.wikipedia.org/wiki/2010_%D0%B3%D0%BE%D0%B4), версия 1.0 была выпущена тремя месяцами позже.

Версия 2.0 вышла [13 декабря](https://ru.wikipedia.org/wiki/13_%D0%B4%D0%B5%D0%BA%D0%B0%D0%B1%D1%80%D1%8F) [2011 года](https://ru.wikipedia.org/wiki/2011_%D0%B3%D0%BE%D0%B4). Версия 3.0 была выпущена [24 сентября](https://ru.wikipedia.org/wiki/24_%D1%81%D0%B5%D0%BD%D1%82%D1%8F%D0%B1%D1%80%D1%8F) [2013 года](https://ru.wikipedia.org/wiki/2013_%D0%B3%D0%BE%D0%B4).

PyCharm Community Edition, бесплатная версия с открытым исходным кодом была опубликована [22 октября](https://ru.wikipedia.org/wiki/22_%D0%BE%D0%BA%D1%82%D1%8F%D0%B1%D1%80%D1%8F) [2013 года](https://ru.wikipedia.org/wiki/2013_%D0%B3%D0%BE%D0%B4).

В [марте](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%80%D1%82) [2016 года](https://ru.wikipedia.org/wiki/2016_%D0%B3%D0%BE%D0%B4) JetBrains перешла на подписную модель лицензирования, а вместе с этим изменилась и нумерация версий. Теперь номер версии выглядит как YYYY.R, где *YYYY* — год выпуска, а *R* — выпуск в течение этого года.

### 2.3.2. Python

Python— [высокоуровневый язык программирования](https://ru.wikipedia.org/wiki/%D0%92%D1%8B%D1%81%D0%BE%D0%BA%D0%BE%D1%83%D1%80%D0%BE%D0%B2%D0%BD%D0%B5%D0%B2%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F) общего назначения, ориентированный на повышение производительности разработчика и читаемости кода. [Синтаксис](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D0%BD%D1%82%D0%B0%D0%BA%D1%81%D0%B8%D1%81_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) ядра Python минималистичен. В то же время [стандартная библиотека](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D0%B0%D0%BD%D0%B4%D0%B0%D1%80%D1%82%D0%BD%D0%B0%D1%8F_%D0%B1%D0%B8%D0%B1%D0%BB%D0%B8%D0%BE%D1%82%D0%B5%D0%BA%D0%B0_Python) включает большой объём полезных функций.

Python поддерживает  [структурное](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5), [функциональное](https://ru.wikipedia.org/wiki/%D0%A4%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D0%BE%D0%BD%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5), [императивное](https://ru.wikipedia.org/wiki/%D0%98%D0%BC%D0%BF%D0%B5%D1%80%D0%B0%D1%82%D0%B8%D0%B2%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) и [аспектно-ориентированное программирование](https://ru.wikipedia.org/wiki/%D0%90%D1%81%D0%BF%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5). Основные архитектурные черты — [динамическая типизация](https://ru.wikipedia.org/wiki/%D0%94%D0%B8%D0%BD%D0%B0%D0%BC%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D1%82%D0%B8%D0%BF%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F), [автоматическое управление памятью](https://ru.wikipedia.org/wiki/%D0%A1%D0%B1%D0%BE%D1%80%D0%BA%D0%B0_%D0%BC%D1%83%D1%81%D0%BE%D1%80%D0%B0_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)), полная [интроспекция](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D1%80%D0%BE%D1%81%D0%BF%D0%B5%D0%BA%D1%86%D0%B8%D1%8F_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)), механизм [обработки исключений](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B0_%D0%B8%D1%81%D0%BA%D0%BB%D1%8E%D1%87%D0%B5%D0%BD%D0%B8%D0%B9), поддержка [многопоточных вычислений](https://ru.wikipedia.org/wiki/%D0%9C%D0%BD%D0%BE%D0%B3%D0%BE%D0%BF%D0%BE%D1%82%D0%BE%D1%87%D0%BD%D0%BE%D1%81%D1%82%D1%8C), высокоуровневые [структуры данных](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80%D0%B0_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85). Поддерживается разбиение программ на [модули](https://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B4%D1%83%D0%BB%D1%8C_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)), которые, в свою очередь, могут объединяться в пакеты.

Эталонной реализацией Python является интерпретатор [CPython](https://ru.wikipedia.org/wiki/CPython" \o "CPython), поддерживающий большинство активно используемых платформ. Он распространяется под [свободной лицензией](https://ru.wikipedia.org/wiki/%D0%A1%D0%B2%D0%BE%D0%B1%D0%BE%D0%B4%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5) Python Software Foundation License, позволяющей использовать его без ограничений в любых приложениях, включая [проприетарные](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%BF%D1%80%D0%B8%D0%B5%D1%82%D0%B0%D1%80%D0%BD%D0%BE%D0%B5_%D0%9F%D0%9E). Есть [реализация интерпретатора для JVM](https://ru.wikipedia.org/wiki/Jython) с возможностью [компиляции](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D0%B8%D0%BB%D1%8F%D1%82%D0%BE%D1%80), [CLR](https://ru.wikipedia.org/wiki/IronPython), [LLVM](https://ru.wikipedia.org/wiki/LLVM), другие независимые реализации. Проект [PyPy](https://ru.wikipedia.org/wiki/PyPy" \o "PyPy) использует [JIT-компиляцию](https://ru.wikipedia.org/wiki/JIT-%D0%BA%D0%BE%D0%BC%D0%BF%D0%B8%D0%BB%D1%8F%D1%86%D0%B8%D1%8F), которая значительно увеличивает скорость выполнения Python-программ.

Python — активно развивающийся [язык программирования](https://ru.wikipedia.org/wiki/%D0%AF%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F), новые версии с добавлением/изменением языковых свойств выходят примерно раз в два с половиной года. Язык не подвергался официальной стандартизации, роль стандарта де-факто выполняет [CPython](https://ru.wikipedia.org/wiki/CPython" \o "CPython), разрабатываемый под контролем автора языка.

### . MongoDB Server

MongoDB — [документоориентированная](https://ru.wikipedia.org/wiki/%D0%94%D0%BE%D0%BA%D1%83%D0%BC%D0%B5%D0%BD%D1%82%D0%BE%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%B0%D1%8F_%D0%A1%D0%A3%D0%91%D0%94" \o "Документоориентированная СУБД) [система управления базами данных](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D1%83%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F_%D0%B1%D0%B0%D0%B7%D0%B0%D0%BC%D0%B8_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85) (СУБД) с открытым исходным кодом, не требующая описания схемы таблиц. Классифицирована как [NoSQL](https://ru.wikipedia.org/wiki/NoSQL" \o "NoSQL), использует [JSON](https://ru.wikipedia.org/wiki/JSON)-подобные документы и схему базы данных. Написана на языке [C++](https://ru.wikipedia.org/wiki/C%2B%2B).

Система может работать с набором [реплик](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%BF%D0%BB%D0%B8%D0%BA%D0%B0%D1%86%D0%B8%D1%8F_(%D0%B2%D1%8B%D1%87%D0%B8%D1%81%D0%BB%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D1%82%D0%B5%D1%85%D0%BD%D0%B8%D0%BA%D0%B0)), то есть, содержать две или более копии данных на различных узлах. Каждый экземпляр набора реплик может в любой момент выступать в роли основной или вспомогательной реплики. Все операции записи и чтения по умолчанию осуществляются с основной репликой. Вспомогательные реплики поддерживают в актуальном состоянии копии данных. В случае, когда основная реплика дает сбой, набор реплик проводит выбор, который из реплик должен стать основным. Второстепенные реплики могут дополнительно являться источником для операций чтения.

Система [масштабируется горизонтально](https://ru.wikipedia.org/wiki/%D0%93%D0%BE%D1%80%D0%B8%D0%B7%D0%BE%D0%BD%D1%82%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D0%B5_%D0%BC%D0%B0%D1%81%D1%88%D1%82%D0%B0%D0%B1%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5), используя технику [сегментирования](https://ru.wikipedia.org/w/index.php?title=%D0%A1%D0%B5%D0%B3%D0%BC%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5&action=edit&redlink=1) объектов баз данных — распределение их частей по различным узлам кластера. Администратор выбирает ключ сегментирования, который определяет, по какому критерию данные будут разнесены по узлам (в зависимости от значений хэша ключа сегментирования). Благодаря тому, что каждый узел кластера может принимать запросы, обеспечивается [балансировка нагрузки](https://ru.wikipedia.org/wiki/%D0%91%D0%B0%D0%BB%D0%B0%D0%BD%D1%81%D0%B8%D1%80%D0%BE%D0%B2%D0%BA%D0%B0_%D0%BD%D0%B0%D0%B3%D1%80%D1%83%D0%B7%D0%BA%D0%B8).

# Программная реализация

В данном разделе будет рассмотрена архитектура приложения, а также будут приведены примеры некоторых пользовательских функций и хранимых процедур.

## Физическая структура приложения

При разработке программного продукта было решено разделить всю основную логику на модули. В целом проект состоит из пяти интегрируемых модулей, каждый из которых выполняет определенную задачу:

* **main\_page.py** – головной файл для работы с приложением. Содержит в себе точку входа и основной пользовательский интерфейс данного проекта. Так же служит для связи между всеми составляющими проекта, включая другие модули и статические классы;
* **mongoDB.py** – файл для работы с базой данных. Состоит из статического класса, который содержит перечень методов для работы с MongoDB и так же проверки этих методов и отслеживанием текущего состояния базы данных;
* **basics.py** – файл с основными рабочими классами приложения. Состоит из таких глобальных классов, как User, Book, Statistics. Помогает вести отчет, отслеживать и работать с текущим пользователем или книгой;
* **machine\_learning.py** – файл для работы с методами машинного обучения. Дополняет проект приятным своеобразным приветствием;
* **patterns.py** – файл для проверки и работы с паттернами, которые внедряются в программу путем включения данного файла в основной проект.

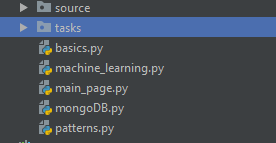


Рис. 2. Составляющие модули приложения

## Архитектура приложения

Приведем пару примеров из архитектурной логики приложения.

На протяжении всего периода своего жизненного цикла приложение обменивается данными со встроенной базой данных, которая, в свою очередь, меняется информацией с основными рабочими классами, прописанными в файле basics.py. К основным относятся следующие классы:

Класс User:

**class** User:  
 user\_id = 0  
 name = **'unknown'** password = **'123'** email = **'unknown@gmail.com'** books = **''** coins = 0  
 image = **'question.png'**  
  
 **def** set(self, user\_id, name, password, email, books, coins, image):  
 self.user\_id = user\_id  
 self.name = name  
 self.password = password  
 self.email = email  
 self.books = books  
 self.coins = coins  
 self.image = image

Класс Statistics:

**class** Statistics:  
 info\_id = 1  
 total\_downloads = 0  
 total\_users = 0  
 total\_earned = 50  
  
 **def** set(self, info\_id, total\_downloads, total\_users, total\_earned):  
 self.info\_id = info\_id  
 self.total\_downloads = total\_downloads  
 self.total\_users = total\_users  
 self.total\_earned = total\_earned  
  
 **def** clear(self):  
 self.info\_id = 0  
 self.total\_downloads = 0  
 self.total\_users = 0  
 self.total\_earned = 0

Класс Book:

**class** Book:  
 title = **'unknown'** author = **'unknown'** price = 0  
  
 **def** set(self, title, author, price):  
 self.title = title  
 self.author = author  
 self.price = price  
  
 **def** get(self):  
 **return** self.title, self.author, self.price

Классы, описанные выше, как уже упоминалось, обмениваются информацией с базой данной, написанной в файле mongoDB.py. Это довольно удобно с точки зрения разработчика, так как не вызывает лишних затрат времени и памяти. В файле mongoDB.py всего один статический класс, но он содержит в себе все ключевые методы для работы с NoSQL DB.

Ниже приведен код, который описывает класс DataBase, включая все его методы и комментарии, необходимые для понимания кода другими людьми:

**class** DataBase:  
 @staticmethod  
 **def** bases\_list():  
 print(myclient.list\_database\_names())  
  
 @staticmethod  
 **def** is\_db\_exist(name):  
 dblist = myclient.list\_database\_names()  
 **if** name **in** dblist:  
 print(**"The database exists."**)  
 **else**:  
 print(**"The database is not exists."**)  
  
 @staticmethod  
 **def** add(collection\_name, value): *# добавлять нужно по типу: { "name": "John", "address": "Highway 37" }* x = mydb[collection\_name].insert\_one(value)  
 print(x)  
  
 @staticmethod  
 **def** add\_many(collection\_name, values):  
 x = mydb[collection\_name].insert\_many(values)  
 print(x)  
  
 @staticmethod  
 **def** delete\_one(collection\_name, value): *# удалять нужно по типу: {"address": "Mountain 21"}* mydb[collection\_name].delete\_one(value)  
  
 @staticmethod  
 **def** delete\_all(collection\_name):  
 mydb[collection\_name].drop()  
  
 @staticmethod  
 **def** find(collection\_name, any\_value): *# искать нужно по типу: { "address": "Park Lane 38" }* **return** mydb[collection\_name].find(any\_value)  
  
 @staticmethod  
 **def** update\_one(collection\_name, value\_to\_change, new\_value):  
 mydb[collection\_name].update\_one(value\_to\_change, new\_value) *# В ЗАМЕНЕ ВСЕГДА ИСКАТЬ ТОЛЬКО ПО ID!!!!!!!*@staticmethod  
 **def** show(collection\_name):  
 **for** i **in** mydb[collection\_name].find():  
 print(i)  
  
 @staticmethod  
 **def** pop(collection\_name):  
 **return** mydb[collection\_name].find() *# для извлечения определенных полей*

Теперь, когда основная логика головных классов и базы данных известна, можно приступать к изучению всех основных имеющихся методов в приложении, которые написаны в основном файле проекта main\_page.py.

Ниже приведен список подключаемых файлов и библиотек, необходимых для корректной работы приложения:

**import** tkinter **as** tk  
**from** tkinter **import** ttk  
**from** tkinter **import** messagebox  
**from** PIL **import** ImageTk, Image  
**import** re *# регулярные выражения***import** winsound *# сигнал об ошибке***import** mongoDB  
**import** basics  
**import** machine\_learning

Головной файл состоит из нескольких объемных классов – страниц – необходимых для корректного отображения каждого модуля, который есть в программе. Приведем пример входа в систему пользователем:

**def** log\_in(self, event):  
 **for** row **in** db.pop(**'users'**):  
 **if** row[**'email'**] == self.entry\_email.get() **and** row[**'password'**] == self.entry\_pass.get():  
 current\_user.set(row[**'\_id'**], row[**'name'**], row[**'password'**], row[**'email'**],  
 row[**'books'**], row[**'coins'**], row[**'image'**])  
 **break  
  
 if** current\_user.user\_id == 0:  
 messagebox.showerror(**"Title"**, **"Неверная почта или пароль"**)  
 **else**:  
 self.hello\_user()  
 app.update\_()  
 self.destroy()

По такому принципу построена взаимосвязь между базой данных и основными классами. Ниже приведен код метода ригестрации:

**def** sing\_in(self, event):  
 **if** self.entry\_name.get() == **'' or** self.entry\_pass.get() == **'' or** self.entry\_email.get() == **''**:  
 self.description.config(text=**'Присутствуют пустые поля'**, fg=**'red'**, anchor=**'center'**)  
 self.pass\_label2.config(fg=**'black'**)  
 self.email\_label.config(fg=**'black'**)  
 **return** is\_valid = re.findall(**'(^|\s)[-a-z0-9\_.]+@([-a-z0-9]+\.)+[a-z]{2,6}(\s|$)'**, self.entry\_email.get())  
 **if not** is\_valid:  
 self.description.config(text=**'Неверный формат почты'**, fg=**'red'**, anchor=**'center'**)  
 self.pass\_label2.config(fg=**'black'**)  
 self.email\_label.config(fg=**'red'**)  
 **return** counter = 1  
 is\_exist = **True  
 if** self.entry\_pass.get() == self.entry\_pass2.get():  
 **for** row **in** db.pop(**'users'**):  
 counter += 1  
 **if** row[**'email'**] == self.entry\_email.get():  
 self.description.config(text=**'Эта почта уже занята'**, fg=**'red'**, anchor=**'center'**)  
 self.email\_label.config(fg=**'red'**)  
 self.pass\_label2.config(fg=**'black'**)  
 is\_exist = **False  
 pass  
 if** is\_exist:  
 db.add(**'users'**, {**'\_id'**: counter, **'name'**: self.entry\_name.get(), **'password'**: self.entry\_pass.get(),  
 **'email'**: self.entry\_email.get(), **'books'**: **''**, **'coins'**: 20, **'image'**: **'boy.png'**})  
 current\_user.set(counter, self.entry\_name.get(), self.entry\_pass.get(), self.entry\_email.get(),  
 **''**, 20, **'boy.png'**)  
 self.update\_users\_statistics()  
 app.update\_()  
 self.destroy()  
 **else**:  
 self.description.config(text=**'Пароли не сопадают'**, fg=**'red'**, anchor=**'center'**)  
 self.pass\_label2.config(fg=**'red'**)  
 self.email\_label.config(fg=**'black'**)

Так же стоит отметить, что в коде имеется машинное обучение, которое тоже упоминалось ранее. Метод работает при помощи встроенной библиотеки sklearn и дает возможность предугадать личность пользователя своеобразным приветствием при входе в систему. Строчки кода, описывающие принцип работы встроенного машинного обучения:

**def** learning(image, value):  
 features = [[0, 2.0], [0, 2.4], [0, 2.9], [0, 3.5], [1, 2.0], [1, 2.4], [1, 2.9], [1, 3.5]]  
 labels = [0, 0, 1, 1, 2, 2, 3, 3]  
  
 classif = tree.DecisionTreeClassifier()  
 classif.fit(features, labels)**return** classif.predict([[image, value]])

А теперь пример кода вызова метода в классе Main:

@staticmethod  
**def** hello\_user():  
 **if** current\_user.books != **''**:  
 app.success\_label.config(text=**'Добро пожаловать, '** +  
 machine\_learning.prediction(current\_user.books, current\_user.image) + **' ^\_^'**)  
 **else**:  
 app.success\_label.config(text=**'Добро пожаловать, '** + current\_user.name)  
 winsound.Beep(300, 200)  
 winsound.Beep(450, 200)

# Описание применения

Помимо различных файлов, приложение состоит из нескольких страниц для взаимодействия с пользователем: Main, Catalog, Account, Registration, Info, Transfer и Statistics. Опишем работу с каждым из них.

## Main

Модуль представлен двумя формами: форма для обычного пользователя (Рис.3) и форма для администратора (Рис.4).

На данной странице пользователь может видеть список купленной литературы, кнопку своего аккаунта, кнопку помощи и кнопку каталога. Администратор видит две дополнительные кнопки: Статистика и Перевод. Только привилегированный пользователь может проводить подобные операции.

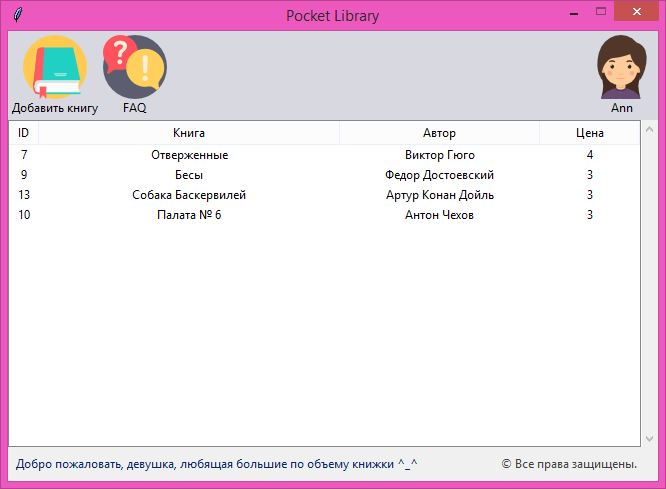


Рис. 4. Страница обычного пользователя

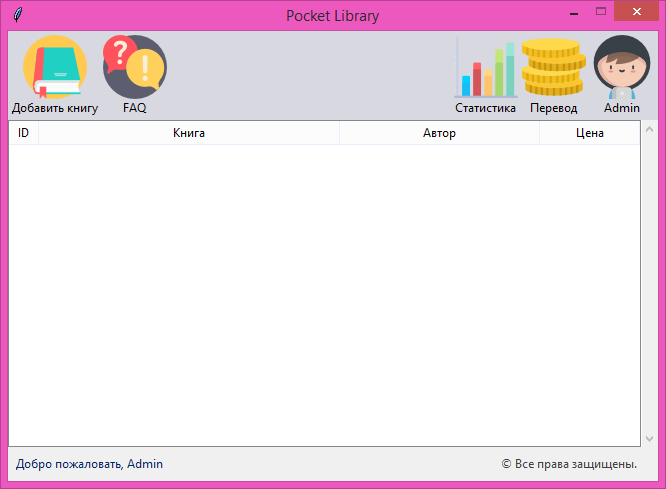


Рис. 5. Страница администратора

## Catalog

Каталог (Рис.6) служит списком для имеющихся в базе данных книг. Там пользователь может приобретать книги за коины. В случае нехватки коинов пользователь получит сообщение об ошибке.

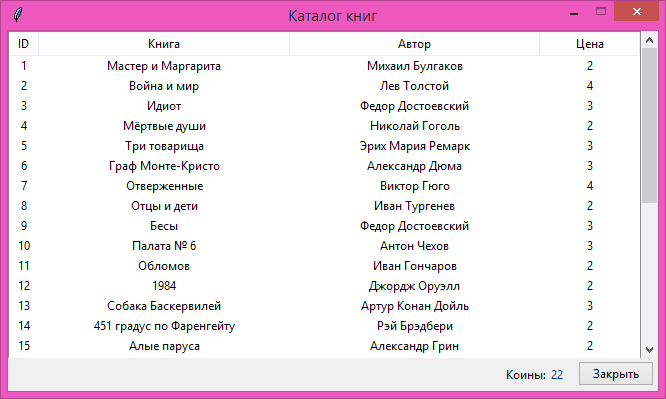


Рис. 6. Страница каталога

## Account/Registration

На данной странице пользователь может посмотреть информацию о своем аккаунте: свои личные данные и количество коинов (Рис.7). Так же можно поменять картинку. В случае если пользователь не вошел в систему, будет предложен вход или регистрация (Рис. 8).

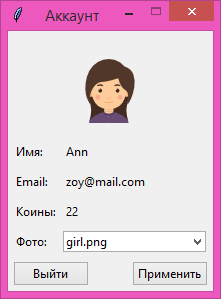


Рис. 7. Аккаунт пользователя

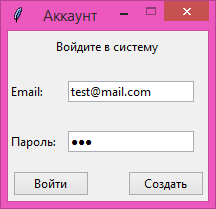
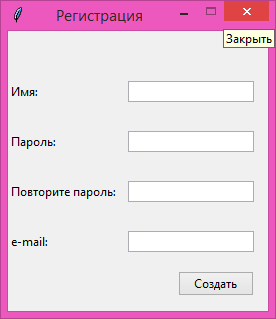
 

Рис. 8. Страницы входа и регистрации

## Info

Страница, предназначенная для пользователя, впервые запустившего приложение. В ней описаны все основные принципы работы с продуктом на понятном для любого человека языке (Рис.9).

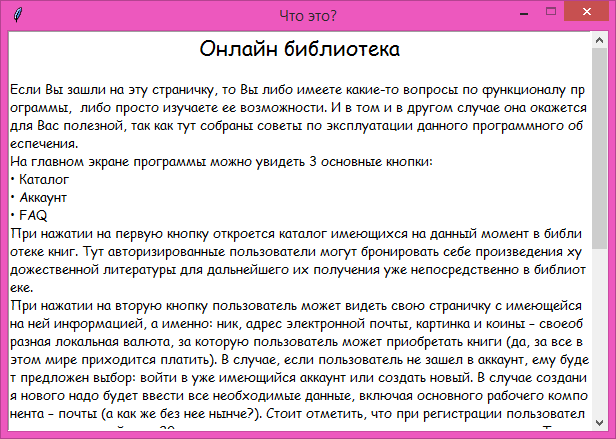
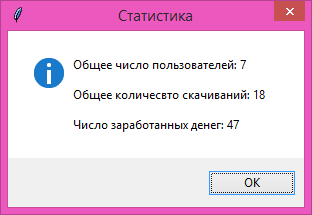
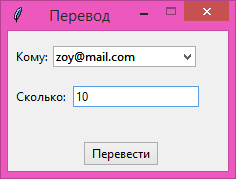


Рис. 9. Страница вопросов и ответов

## Transfer/Statistics

Страницы, которые может видеть только администратор. Страница Statistics позволяет увидеть текущую статистику приложения. Страница Transfer позволяет перевести коины любому пользователю (Рис.10).

  Рис. 10 Страницы статистики и перевода

# Заключение

В рамках работы над курсовым проектом был разработан программный продукт для управления работой онлайн библиотекой.

За время написания проекта была изучена информация о принципах работы библиотек и подходы решения основных задач, возлагаемых на администрацию библиотеки.

В плане разработки приложения был изучен большой объем информации по проектированию и администрированию баз данных, углублены и закреплены знания по разработке приложений на платформе Python Tkinter.

Были проанализированы различные подходы и технологии, из которых были выбраны те, которые автор хотел бы видеть в своем программном продукте.

Разработанное программное средство представляет собой законченный программный продукт, готовый к использованию. Но при желании приложение можно доработать: расширить функциональность приложения, изменить дизайн и др.

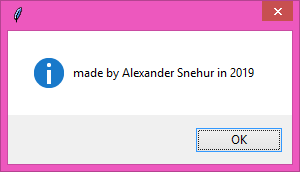


Рис. 11 Информация об авторе

# Список использованных источников

1. <https://ru.wikipedia.org/wiki/Python>
2. <https://metanit.com/python/tutorial/9.1.php>
3. <https://www.w3schools.com/python/python_mongodb_getstarted.asp>
4. <https://habr.com/ru/post/319288/>
5. <https://techrocks.ru/2018/10/05/python-libraries-for-machine-learning/>
6. <https://en.wikipedia.org/wiki/MongoDB>
7. <https://pythonworld.ru/>

# Приложения

Конструктор класса Main:

*# конструктор***def** \_\_init\_\_(self, some\_root):  
 super().\_\_init\_\_(some\_root)  
  
 self.success\_label = tk.Label(root, text=**''**, fg=**'#002966'**)  
 self.main\_label = tk.Label(root, text=**'© Все права защищены.'**, fg=**'#404040'**)  
 self.tree = ttk.Treeview(self, columns=(**'id'**, **'book'**, **'author'**, **'price'**),  
 height=15, show=**'headings'**, selectmode=**'browse'**)  
  
 *# 64x64, colored only!!* self.add\_book\_img = ImageTk.PhotoImage(Image.open(sources + **"book.png"**))  
 self.add\_user\_img = ImageTk.PhotoImage(Image.open(sources + **"question.png"**))  
 self.add\_info\_img = ImageTk.PhotoImage(Image.open(sources + **"faq.png"**))  
  
 self.toolbar = tk.Frame(bg=**"#d7d8e0"**, bd=2)  
 self.account\_button = tk.Button(self.toolbar, text=**'Войти'**, command=self.open\_account,  
 bg=**'#d7d8e0'**, bd=0, compound=tk.TOP, image=self.add\_user\_img)  
 self.catalog\_button = tk.Button(self.toolbar, text=**'Добавить книгу'**, command=self.open\_catalog, bg=**'#d7d8e0'**,  
 bd=0, compound=tk.TOP, image=self.add\_book\_img)  
 self.info\_button = tk.Button(self.toolbar, text=**'FAQ'**, command=self.open\_info,  
 bg=**'#d7d8e0'**, bd=0, compound=tk.TOP, image=self.add\_info\_img)  
 self.statistics\_button = tk.Button(self.toolbar, text=**'Статистика'**, command=self.open\_statistics, bg=**'#d7d8e0'**,  
 bd=0, compound=tk.TOP, image=self.add\_statistics\_img)  
 self.transfer\_button = tk.Button(self.toolbar, text=**'Перевод'**, command=self.open\_transfer, bg=**'#d7d8e0'**,  
 bd=0, compound=tk.TOP, image=self.add\_transfer\_img)  
  
 self.init\_main()  
 self.db = db *# для обращения к методам из DataBase* self.current\_user = current\_user *# для получения информации из класса User* self.current\_statistics = current\_statistics *# для получения информации из класса Statistics*

Создание графических объектов на главной странице:

*# хранит и инициализирует все объекты графического интерфейса***def** init\_main(self):  
 self.create\_toolbar()  
 self.create\_tree()  
 self.main\_label.place(x=490, y=422)  
 self.success\_label.place(x=5, y=422)  
 self.main\_label.bind(**'<Double-3>'**, self.easter\_egg)  
  
 self.load\_statistics()  
  
**def** create\_toolbar(self):  
 self.toolbar.pack(side=tk.TOP, fill=tk.X)  
  
 self.catalog\_button.pack(side=tk.LEFT)  
 self.account\_button.pack(side=tk.RIGHT)  
 self.info\_button.pack(side=tk.LEFT)  
  
**def** create\_tree(self):  
 *# создание таблицы на главной* self.tree.column(**'id'**, width=30, anchor=tk.CENTER)  
 self.tree.column(**'book'**, width=300, anchor=tk.CENTER)  
 self.tree.column(**'author'**, width=200, anchor=tk.CENTER)  
 self.tree.column(**'price'**, width=100, anchor=tk.CENTER)  
  
 self.tree.heading(**'id'**, text=**'ID'**)  
 self.tree.heading(**'book'**, text=**'Книга'**)  
 self.tree.heading(**'author'**, text=**'Автор'**)  
 self.tree.heading(**'price'**, text=**'Цена'**)  
  
 scrollbar = ttk.Scrollbar(self, orient=**"vertical"**, command=self.tree.yview)  
 self.tree.pack(side=**'left'**) *# отображение таблицы* scrollbar.pack(side=**'right'**, fill=**'y'**)  
 self.tree.configure(yscroll=scrollbar.set)